Tutoriel sur la manipulation des données d'affaires – Qu'est-ce qu'un environnement virtuel (venv) : pourquoi et comment l'utiliser?

Introduction et contexte de la problématique d'affaires :

Dans le domaine en constante évolution de la science des données, la manipulation efficace des données d'affaires est essentielle pour les professionnels et les entreprises. L'un des outils fondamentaux pour gérer les dépendances et isoler les environnements de développement est l'utilisation d'environnements virtuels en Python, souvent nommés venv. Dans ce tutoriel, nous allons explorer ce qu'est un environnement virtuel, pourquoi il est important et comment le créer.

Un environnement virtuel est un espace d'exécution isolé qui permet de gérer les dépendances d'un projet de manière indépendante. Autrement dit, il permet d'isoler les paquets* utilisés pour un projet, afin d'avoir des versions différentes pour chaque projet.

La gestion des versions de paquets pour différents projets de données d'affaires est un réel défi rencontré par les entreprises et les professionnels. L'utilisation d'un environnement virtuel comprend plusieurs avantages que nous allons voir au fil et à mesure dans ce tutoriel, dont l'isolation des dépendances, la reproductibilité, une meilleure gestion des versions, un nettoyage facile ainsi qu'une meilleure portabilité.

*Note
Un paquet est un ensemble de plusieurs modules regroupés entre eux. Peut aussi avoir le nom de package ou bibliothèque.
Un module est tout fichier constitué de code Phyton (avec l'extension .py) importé dans un autre fichier ou script.

Connaissances préalables :

Afin de suivre ce tutoriel, vous devriez avoir une connaissance de base en Phyton, comprendre les structures de données en Phyton et être familier avec les concepts de gestion de dépendances.

Exemple mise en situation :

Prenons un exemple concret pour illustrer l'importance des environnements virtuels. Lorsque vous travaillez sur plusieurs projets Phyton, chaque projet peut nécessiter des paquets spécifiques avec des versions différentes.

Par exemple, un site web de commerce électronique en pleine croissance peut utiliser Django 1.3, puis un autre projet d'analyse de données visant à améliorer les stratégies de marketing basées sur les données peut nécessiter Django 1.0. Donc, le site web peut nécessiter des fonctionnalités avancées introduites dans Django 1.3, puis le projet d'analyse de données peut être basé sur un code existant qui n'est pas compatible dans Django 1.3.

Ainsi, puisqu'il y a des incompatibilités entre les deux versions de Django, cela pourrait entraîner des erreurs de fonctionnement, des failles de sécurité ou des retards dans le développement, ce qui aurait un impact négatif direct sur les objectifs des projets.

Dans ce cas, pour éviter les conflits entre ces deux projets et garantir la compatibilité et la cohérence des dépendances, l'utilisation d'un environnement virtuel est indispensable pour les développeurs qui désirent travailler efficacement sur plusieurs projets en même temps.

Comment créer et utiliser un environnement virtuel dans Python?

1ère étape : Créer un environnement virtuel :

Commande : python -m venv nom_environnement

Ici, « nom_environnement » est le nom que vous donnez à ce nouvel environnement virtuel. Un nouveau dossier portant ce nom sera alors créé pour contenir les dépendances du projet.

2e étape : Activer l'environnement virtuel :

Commande sur Windows : nom_environnement\Scripts\activate
Commande sur MacOs/Linux : source nom_environnement/bin/activate

Ici, votre environnement virtuel sera activé et par le fait même, vous isolez les dépendances de votre projet.

3e étape : Installer les dépendances :

Commande : pip install Django

Ici, cette fonction permet d'installer la dernière version d'un paquet en indiquant son nom. Si vous voulez installer une version spécifique d'un paquet, il faut utiliser ce code :
Commande : pip install Django = = 1.3

4e étape : Générer un fichier « requirements.txt. :

Commande : pip freeze > requirements.txt

Après avoir installé toutes les dépendances (étape 3), vous pouvez générer le fichier « requirements.txt » en exécutant la commande ci-dessus. Celle-ci générera un fichier « requirements.txt. » contenant la liste de tous les paquets installés, ainsi que leurs versions spécifiques. Ce fichier pourra être utilisé pour installer les mêmes dépendances dans un autre environnement virtuel ou pour simplement partager votre liste de dépendances avec des collaborateurs qui travaillent avec vous.

5e étape : Exécuter votre code :

Une fois les dépendances installées, vous pouvez exécuter votre code comme d'habitude.

6e étape : Désactiver l'environnement virtuel :

Commande : deactivate

Désactiver l'environnement virtuel lorsque vous avez terminé de travailler sur votre projet.

Bonnes pratiques et astuces pour la manipulation efficace des données :

Afin d'avoir une gestion efficace de vos environnements virtuels et des dépendances, voici certaines bonnes pratiques à suivre.

Il est important de toujours utiliser un environnement virtuel pour chaque projet Phyton. De plus, pour une reproductibilité maximale, il peut être intéressant de sauvegarder les dépendances dans un fichier nommé « requirements.txt.» dont nous avons vu précédemment dans ce tutoriel. Il est aussi important de désactiver l'environnement virtuel lorsque vous avez terminé de travailler sur votre projet, dans le but d'éviter les conflits avec d'autres projets.

Diverses applications des environnement virtuel :

Comme nous avons vu, l'utilisation des environnements virtuels est nécessaire lorsque vous travaillez sur des projets ayant des paquets différents, comme Django et Pandas par exemple, ou même des versions différentes d'un même paquet comme nous l'avons vu dans l'exemple de mise en situation, en utilisant Django 1.3 et Django 1.0. Les environnements virtuels peuvent donc être utilisés pour isoler des environnements de travail de différents projets, pour isoler les environnements de développement et pour isoler les paquets et les ressources utilisées dans le processus de développement.

De plus, les projets en open source peuvent bénéficier des environnements virtuels car ceux-ci leur permettent de spécifier et de gérer les dépendances nécessaires pour exécuter le projet. En effet, dans un projet open source, il est possible que de nombreux contributeurs soient impliqués, chacun ayant son propre environnement de développement et ses propres versions de paquets. Donc, en utilisant des environnements virtuels, les développeurs peuvent spécifier exactement quelles versions des dépendances sont nécessaires pour le projet, assurant ainsi que les contributeurs travaillent avec les mêmes versions de paquets.

Un autre cas d'utilisation peut être dans le cadre d'un cours à l'université par exemple, où les environnements virtuels sont utilisés pour fournir un environnement de développement cohérent et isolé à chaque étudiant. Chaque étudiant a la possibilité de créer son propre environnement virtuel qui inclut les versions spécifiques de Phyton et les paquets nécessaires pour suivre le cours. L'utilisation des environnements virtuels permet donc d'assurer aux étudiants une expérience d'apprentissage uniforme et reproductible, quelles que soient les différences dans leurs versions de logiciels.

Les limites et les prochaines étapes :

Malgré l'efficacité des environnements virtuels pour gérer les dépendances dans de nombreux cas, il existe des situations où ils peuvent comporter des limites. En effet, pour les projets avec de nombreuses dépendances ou lorsque celles-ci sont plus complexes, la gestion des environnements virtuels peut être plus complexe surtout si plusieurs environnements sont nécessaires. Aussi, la taille des environnements virtuels peut également entraîner une surcharge de stockage si plusieurs versions de paquets sont installées pour différents projets. Finalement, il est important d'être vigilant avec les mises à jour des paquets pour ne pas créer d'autres problèmes.

Pour pousser vos apprentissages encore plus loin et connaître quelles sont les prochaines étapes avec les environnements virtuels, vous pouvez vous informer sur comment utiliser des outils d'automatisation pour simplifier la création, la gestion et la suppression des environnements virtuels en utilisant des scripts. Aussi, il est possible d'intégrer des environnements virtuels dans des pipelines d'intégration continue pour assurer la cohérence des environnements de développement tout au long de la durée du projet. Il existe également des manières d'optimiser les performances des environnements virtuels. Finalement, il peut être intéressant de regarder d'autres outils similaires comme par exemple Conda, qui est un gestionnaire de paquets et qui permet la création d'environnements virtuels et puis par la suite, déterminer quelles solutions de gestion des dépendances convient le mieux selon les scénarios de vos projets.

Conclusion :

Pour conclure, plusieurs professionnels et entreprises ont à travailler sur plusieurs projets Phyton simultanément et il est important de savoir comment avoir une gestion efficace des dépendances dans la manipulation des données d'affaires en Phyton. Nous avons vu dans ce tutoriel que l'utilisation de l'environnement virtuel permet de résoudre les problèmes de compatibilité de version, les conflits de dépendances et les erreurs d'exécution. En suivant les étapes mentionnées plus haut dans ce tutoriel, il est très simple et facile de créer et gérer des environnements virtuels pour chacun de vos projets.

Nous avons également vu quelques concepts clés dont l'isolation des dépendances à l'aide d'environnements virtuels, la gestion des dépendances avec la commande pip et les bonnes pratiques à se souvenir pour maintenir la cohérence et la reproductibilité des environnements de développement.

En ayant une bonne maîtrise et une bonne compréhension des environnements virtuels, vous serez mieux équipés pour gérer efficacement les dépendances de vos projets et ainsi, livrer des données de meilleure qualité.

Sources :

https://www.docstring.fr/glossaire/environnement-virtuel/

https://www.pierre-giraud.com/python-apprendre-programmer-cours/module-paquet/

https://docs.python.org/fr/3/tutorial/venv.html

https://python-guide-pt-br.readthedocs.io/fr/latest/dev/virtualenvs.html

https://www.thepingouin.com/2023/03/29/python-environnement-virtuel-et-pourquoi-vous-devriez-lutiliser/

https://docs.alliancecan.ca/wiki/Python/fr#Cr%C3%A9er_un_environnement_virtuel_dans_vos_t%C3%A2ches

https://www-calculco.univ-littoral.fr/utilisation/tutoriaux/python-environnement-virtuel

https://www.invivoo.com/environnements-virtuels-en-python/

https://www.syloe.com/glossaire/virtualisation-open-source/